Haaga-Helia
ammattikorkeakoulu Oy

DISCOUNT GRID - hybridin mobiiliohjelman yhdistäminen ulkoiseen tietokantaan

Henri Lindberg

Opinnäytetyö
Tietojenkäsittelyn koulutusohjelma
2016

Haaga-Helia
ammattikorkeakoulu Oy

Tiivistelmä

13.10.2016

Tekijä
Henri Lindberg

Koulutusohjelma
Tietojenkäsittelyn koulutusohjelma

Opinnäytetyön otsikko
DISCOUNT GRID - hybridin mobiiliohjelman yhdistäminen ulkoiseen tietokantaan

Sivu- ja liitesivumäärä
20 + 6

Opinnäytetyön otsikko englanniksi
DISCOUNT GRID - connecting hybrid mobile application to an external database

Projektin tavoitteena oli rakentaa kevään-kesän 2016 aikana, oman ideani pohjalta asiakas-palvelin tyyppinen kokonaisuus, jossa Androidille julkaistava mobiiliohjelma hyödyntäisi ulkoisellla palvelimella olevia resursseja sekä tietokantaa. Kokonaisuuden osina olivat hybriditekniikalla toteutettu mobiiliohjelma, ulkoisella palvelimella sijaitseva kaupallinen tietojen sekä kuvien tallennusjärjestelmä sekä itse tietokanta ja palvelimen tiedostontallennusresurssit. Toteutukseen tarvittiin mm. yhdistelmä Ajaxia, PHP:tä, SQL:ää, PHP:n yhtä monista JSON-funktioista sekä edelleen JavaScriptiä ja CSS:ää. Tutkimusaiheeni kiteytyi siihen, miten ulkoinen tietokantatieto ja ohjelmassa esitettävä rivikohtainen kuva saatiin näytettyä mobiiliohjelmassa, jossa ei ollut käytettävissä PHP-tulkkia.

Projektin lopputuloksena syntyi alkuperäistä suunnitelmaani vastaava mobiiliohjelma, joka kykeni esittämään ulkoisella palvelimella sijaitsevat tiedot ja kyseinen ohjelma julkaistiin yleisön saataville Google Playssa nimellä DISCOUNT GRID.

En ollut koskaan aiemmin tehnyt mobiiliohjelmia, joten oppimisprosessi oli monella tapaa hyödyllinen. Ensimmäisen ohjelman valmistuttua olen tehnyt jo pari muutakin ja uusia ideoita on kehitteillä. Havaitsin konkreettisesti miten valtavaa sekä monipuolista mobiiliohjelmien tarjonta on nykyään eli menestyäkseen markkinoilla, monen asian on oltava kunnossa, ideasta-toteutukseen ja markkinointiin. Jatkossa, tietokantojen hyödyntämisen lisäksi, kiinnostaa varmasti hyödyntää avointa dataa, jonka maailmanlaajuinen tarjonta lisääntyy jatkuvasti.

Asiasanat
Ajax, JavaScript, PHP, MySql, mobiiliohjelma

Haaga-Helia
University of Applied Sciences Ltd.

Abstract

13 October 2016

Author
Henri Lindberg

Degree programme
Business Information Technology

Report/thesis title
DISCOUNT GRID - connecting hybrid mobile application to an external database

Number of pages and appendix pages
20 + 6

Keywords
Ajax, JavaScript, PHP, MySql, mobile application

Sisällys

1 Johdanto

Toteutan opinnäytetyössäni, oman ideani pohjalta HTML5:een pohjautuvan hybriditekniikalla toteutetun mobiiliohjelman, joka hakee esitettävän datan ulkoisesta tietokannasta, tässä tapauksessa omalta kotiserveriltäni. Opinnäytetyöni tarkoituksena on selvittää, miten hybridiin mobiiliohjelmaan voidaan tuoda dynaamisesti päivittyvää tietoa ulkoisesta relaatiotietokannasta, sekä esitellä kyseistä tekniikkaa. En siis tarkastele itse mobiiliohjelman tai ulkoisen tietokannan toteutusta tutkimuskohteena, vaan esittelen tutkimusongelmaan liittyvät kooditason teoriat sekä yhden ratkaisun, joka on yhdistelmä Ajaxia, PHP:tä, SQL:ää, PHP:n yhtä monista JSON-funktioista sekä edelleen JavaScriptiä, jolla haluttu tieto saatetaan mobiiliohjelmassa esitettävään muotoon. Lopullinen ulkoasu esitettävälle tiedolle viimeistellään lopuksi CSS:llä.

Mobiiliohjelma toteutetaan Intel XDK kehitysalustalla ja ohjelma on tarkoitus julkaista ainakin Android-laitteille Google Playssa. Ohjelman ideana on näyttää alennuskoodeja ja muita tarjouksia suoraan tietokannasta eri kategorioiden alla, mutta samalla tekniikalla voitaisiin esitellä myös esimerkiksi yritysten tietokantapohjaisia tuotekatalogeja tai toteuttaa vaikkapa uutispalvelu.

Tämä tutkimusaihe kiinnosti, koska teen työkseni mm. tietokantaohjelmointia www-ympäristössä, jossa PHP-tulkki on aina käytettävissä. Miten siis tietojenhakuoperaatio toteutettaisiin käyttäjän puhelimessa sijaitsevan hybriditekniikkaan perustuvan mobiiliohjelman tarjoamien resurssien sekä ulkoisella palvelimella sijaitsevan relaatiotietokannan välillä? Asiakkaani ovat myös kyselleet tämän tapaisen ratkaisun mahdollisuutta, niin pääsin nyt samalla tutustumaan aiheeseen tarkemmin opinnäytetyön kautta.

Intel XDK on vapaasti ladattavissa ja lisenssiehtojen rajoissa hyödynnettävissä oleva kehitysalusta HTML5 -mobiiliohjelmien kehittämiseen.
Ohjelma on yksi helpoimmista tavoista lähteä toteuttamaan mobiiliohjelmia, koska siihen sisältyy mm. drag-and-drop -tyylinen käyttöliittymäsuunnittelutyökalu, joka käyttää HTML5:ttä sekä sen yleisesti hyödynnettävissä olevia resursseja, kuten JavaScript -kirjastoja sekä tunnettuja CSS -viitekehyksiä kuten Bootstrap, Ionic ja Framework 7. Ohjelmakoodia voi kirjoittaa myös itse JavaScriptillä, toteuttaa omia CSS-muotoiluja mutta esim. PHP-tulkkia siihen ei sisälly. Tekniikka on siis tätä rajoitusta lukuunottamatta tuttu nykyaikaisista web-sivustoista ja tutkimusaiheeni kiteytyykin juuri tähän, eli miten saamme ulkoiset resurssit, tässä tapauksessa datan palvelimella sijaitsevasta relaatiotietokannasta sekä palvelimelle tallennetun kuvan, mobiiliohjelman käyttöön.

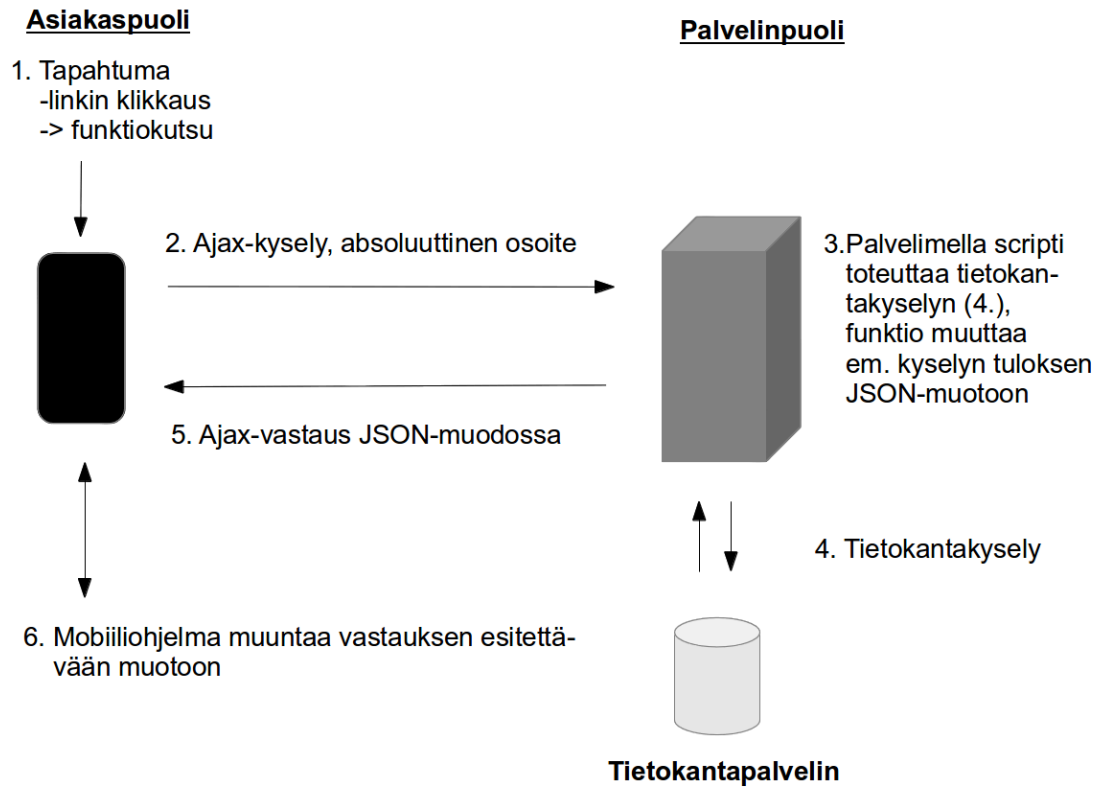
Valmiiksi tehty ohjelma on mahdollista muokata toimivaksi mobiiliohjelmaksi Android, iOS, Windows 8 ja Windows 8.1 -käyttöjärjestelmille. Eräs hybridin mobiiliohjelman etu on, että yhdellä koodilla voidaan toteuttaa ohjelma useammalle alustalle. Intel XDK:n käyttö edellyttää kirjautumista palvelun käyttäjäksi ja se on saatavilla OS X, Windows ja Linux -käyttöjärjestelmille. (Intel XDK:n lisenssiehdot. Luettavissa: https://software.intel.com/xdk/articles/terms-and-conditions-for-intel-xdk. Luettu: 24.4.2016.)
(Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/lp-index. Luettu: 25.4.2016.)

Käsiteluettelo:

Hybridi mobiiliohjelma	HTML5:een, JavaScriptiin ja CSS:ään perustuva mobiiliohjelmien rakennustekniikka, jonka koodi on hyödynnettävissä kaikille mobiilialustoille. Kehitysalusta huolehtii alustaspesifeistä eroavaisuuksista ja tekniikka on ns. järjestelmäriippumaton (engl. cross-platform).
Ajax	Englannikielisistä sanoista Asynchronous JavaScript And XML. Nykyään termiä käytetään laajemmin kuvaamaan kaikkia niitä tekniikoita, joiden avulla selaimen on mahdollista hakea tietoja palvelimelta, ilman tarvetta päivittää koko sivua uudelleen.
CSS	Englanninkielisistä sanoista Cascading Style Cheets. Erityisesti www-dokumenteille kehitetty tyyliohjeiden laji.
SQL	Englannikielisistä sanoista Structured Query Language. IBM:n kehittämä standardoitu kyselykieli, jolla relaatiotietokantaan voi tehdä erilaisia hakuja, muutoksia ja lisäyksiä.
PHP	Laajasti käytössä oleva, dynaamisten web-palveluiden tuottamiseen tarkoitettu ohjelmointikieli. Lyhenne tulee englanninkielisistä sanoista Hypertext Preprocessor.
JSON	Lyhenne sanoista JavaScript Object Notation. Avoimen standardin kevyt tiedostomuoto, jota käytetään mm. järjestelmien väliseen tiedonvälitykseen.
JavaScript	Tavallisimmin selaimessa suoritettava komentosarjakieli.
jQuery	Kaikille selaimille tarkoitettu ilmainen, avoimen lähdekoodin JavaScript-kirjasto.
MySql	Nykyään Oraclen omistama, laajasti käytössä oleva relaatiotietokantaohjelmisto.

2 Havaintokuva tutkimusaiheesta

Asiakas-palvelin tyyppinen toteutus, jossa mobiiliohjelma hakee tietoa ulkoiselta palvelimelta (Kuva 1).



Kuva 1. Havaintokuva tutkimusaiheesta.

3 Ajax

AJAX on lyhenne sanoista Asynchronous JavaScript and XML. Ajax on nimenomaisesti asiakaspuolen tekniikka ja se on myös täysin riippumaton palvelinpuolella käytettävästä tekniikasta. Termiä Ajax käytetään nykyään kuvaamaan yleisimmin kaikkia niitä tekniikoita, joiden avulla selaimen (tai muun asiakaspuolen sovelluksen) on mahdollista hakea tietoja palvelimelta, ilman tarvetta päivittää koko sivua uudelleen. Yhdellä lyhenteellä ”Ajax” ei ole mahdollista kuvata koko tekniikan kattavaa toiminnallisuutta ja esim. XML ei ole lainkaan välttämätön käytettäessa Ajaxia.
(Asleson, R., Scutta, N. T. 2006, 16, 19.)
(W3Schools. AJAX Inroduction. Luettavissa:
http://www.w3schools.com/ajax/ajax_intro.asp. Luettu: 30.4.2016.)

3.1 Katsaus Ajaxin historiaan ja sen standardiin asemaan

Ajax-tekniikan kehittäjästä on käyty keskusteluja mutta termi ”Ajax” mainittiin ensimmäisen kerran helmikuussa 2005, kun Adaptive Pathilla työskennellyt Jesse James Garret julkaisi kirjoituksensa otsikolla Ajax: verkkosovellusten uusi lähestymistapa. Ennen kyseisen termin esittämistä, oli jo vuosien ajan ollut mahdollista käyttää XmlHttpRequest -objektia (XHR) selaimen ja palvelimen väliseen asynkroniseen kommunikointiin. Kyseinen ominaisuus julkaistiin Active-X komponenttina ensimmäisen kerran Internet Explorerin viidennessä versiossa jo vuonna 1999. XHR tuli kuuluisaksi kehittäjäpiireissä Googlen kehitysyksikön, Google Labsin julkaistessa Google Mapsin ja Google Suggestin, joissa tekniikkaa käytettiin. Selaintuen tultua nopeasti myös muihin selaimiin, on XHR nykyään laajasti käytössä myös tunnetuimmissa nettipalveluissa (mm. Gmail, Google Maps, Yahoo, Netflix, Amazon) ja myös noussut de-facto -standardin asemaan. (Asleson, R., Scutta, N. T. 2006. Johdanto, 14, 16)

Kuriositeettina, Ajax ei ole aina ollut täysin standardi menetelmä, vaikka W3C oli standardoinut mm. oliomallit (DOM Level 3 Load and Save Specifikation). Standardointi edellyttäisi, että tekniikka toimisi kaikilla selainalustoilla samalla tavalla. (Asleson, R., Scutta, N. T. 2006. 25-26, 38-39)

W3Schoolsin sivuilla kerrotaan aiheesta ajankohtaisemmin:

> AJAX is based on internet standars, and uses a combination of:
> -XMLHttpRequest object (to retrieve data from a web server) and
> - JavaScript/DOM (to display/use the data)

(W3Schools. AJAX Inroduction. Luettavissa: http://www.w3schools.com/ajax/ajax_intro.asp. Luettu 1.5.2016).

Tämä on nykyään totta, mutta Internet Explorer versiot 5 ja 6 loivat kyseisen XHR-objektin Active-X komponentin avulla, kun muut selaimet ovat luoneet sen normaalina JavaScript-objektina ja kirjassa Ajax – Tehokas hallinta kiinnitetään huomiota siihen, että tämä ei täytä varsinaisesti standardin määritelmää. Vasta Internet Explorerin versiossa 7, XHR-objekti luotiin standardin mukaisena JavaScript-objektina.
(Asleson, R., Scutta, N. T. 2006, 25)
(Mozilla Developer Network. Using XMLHttpRequest in IE6. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest_in_IE6. Luettu: 2.10.2016)

Ajaxia hyödynnetään kuitenkin yhä useammin kirjastojen avulla (lähteessä puhutaan kehyksistä, esim. Ajax-kehys), jotka ottavat automaattisesti tämänkin pienen poikkeaman huomioon, joten ohjelmakehitykselle tämä ei aiheuta suurempia ongelmia. (Asleson, R., Scutta, N. T. 2006. 226, 257-261)

Nykyään XHR-objekti luodaan myös Internet Explorerissa (versiosta 7 alkaen) standardin mukaisessa (window.XMLHttpRequest) JavaScript-objektimuodossa joten kehitys on lopulta johtanut siihen, että kyseessä on myös XHR-objektin muodostamisessa päästy yleisesti noudatettavaan standardiin.
(Asleson, R., Scutta, N. T. 2006. 25)
(W3Schools. AJAX Introduction. Luettavissa:
http://www.w3schools.com/ajax/ajax_intro.asp. Luettu: 30.4.2016.)
(Microsoft. XMLHttpRequest object. https://msdn.microsoft.com/fi-fi/library/ms535874%28v=vs.85%29.aspx. Luettu: 2.5.2016.)

3.2 Ajaxin XHR -objekti

XHR -objektin hyödyntäminen, eli kyselyjen ja vastausten aikaan saaminen, edellyttää XHR -objektin luomista. Koska XMLHttpRequest ei ole ollut standardi menetelmä, sen on voinut luoda kahdella tavalla. Internet Explorer 5 ja 6 toteuttaa objektin Active-X komponentin avulla ja muut selaimet JavaScript-objektina. (Asleson, R., Scutta, N. T. 2006. Johdanto, 25, 26)
(Mozilla Developer Network. Using XMLHttpRequest in IE6. Luettavissa:
https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest_in_IE6. Luettu: 2.10.2016)

Selain- eli asiakaspuolella voitaisiin tehdä seuraavanlainen ns. pseudokoodin mukainen tarkistus, joka kuvaa hyvin tätä aiempaa, epästandardia tilannetta ja joka on johdettu suoraan seuraavan kappaleen kirjan esimerkistä:

```
function tee XHR-kysely()
        jos (selain käyttää Active-X komponentteja)
        Luodaan objekti Active-X komponentin avulla
        muuten (selain käyttää JavaScript-objektia)
        Luodaan XHR-objekti JavaScript-objektina
```

Sama lyhyt koodi syntaksin mukaisessa muodossa (suora lainaus Asleson, R., Scutta, N. T. 2006. 26):

```
var xmlHttp;
function createXMLHttpRequest () {
if (window.ActiveXObject) {
xmlHttp = new ActiveXObject(”Microsoft.XMLHTTP”);
}
else if (window.XMLHttpRequest) {
xmlHttp = new XMLHttpRequest();
}
}
```

Nykyään vanhat Internet Explorer versiot, jotka hyödynsivät Active-X-objektia, (versiot 5 ja 6) alkavat olla jo poistuneet käytöstä.

3.3 Ajaxin turvallisuuteen liittyviä rajoituksia

Kirjassa Ajax – Tehokas hallinta, otetaan kantaa XMLHttpRequestin käytön rajoituksiin turvallisuusnäkökohdista:

> ”XMLHttpRequest-objekti toimii selaimen turvallisen ”hiekkalaatikon” sisällä, jolloin jokaisen XMLHttpRequest-objektin pyytämän resurssin on sijaittava saman verkkotunnuksen (domain) sisällä, josta skriptin suoritus on aloitettu. Tämä turvallisuusrajoite estää XMLHttpRequest-objektia vastaamasta alkuperäisen verkkotunnuksen ulkopuolisiin pyyntöihin.”

Lisäksi, eri selaimet reagoivat tähän ns. ”Same-Origin-Policy”:n rajoitukseen eri tavoin.

> ”Turvallisuusrajoitteiden vahvuus vaihtelee selaimittain. Internet Explorer varoittaa käyttäjää mahdollisesta turvallisuusuhan olemassaolosta, mutta antaa käyttäjälle mahdollisuuden jatkaa pyynnön suorittamista eteenpäin. Firefox yksinkertaisesti keskeyttää pyynnön suorittamisen ja tulostaa JavaScript-konsoliin virheilmoituksen.”

(Asleson, R., Scutta, N. T. 2006. 37)
(Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/how-to-access-JSON-data-in-HTML5-apps. Luettu: 3.5.2016.)

Kirjan JSON-formaattia käsittelevässä luvussa ei oteta kantaa ko. formaattiin kohdistuviin poikkeuksiin (kuten Intel XDK:n dokumentaatiossa), mutta kappaleessa Web Services -palveluiden käyttö (s.117-118) kerrotaan että em. Firefoxin rajoituksen voi kiertää sille räätälöidyllä JavaScriptillä sekä tarjotaan myös yhdyskäytävän rakentamista yhtenä

ratkaisuna. Kirjassa tiedostetaan että kehitys Ajaxin kohdalla on nopeaa, ja koska kirja on alkuperäisenä englanninkielisenä versiona julkaistu vuonna 2006, on selvää että kehitys on ajankohdan jälkeen mennyt huimasti eteenpäin. Edelleenkin, tämä ns. ”Same-Origin-Policy” on olemassa, mutta siihen liittyvät turvallisuuskysymykset ovat saaneet mm. JavaScript-kirjastojen myötä osakseen kehittyneempiä ratkaisuja.
(Asleson, R., Scutta, N. T. 2006. Johdanto, 31, 37, 117-118)
(JQuery Ajax-funktion dokumentaatio: http://api.jquery.com/jquery.ajax/ Luettu: 3.5.2016.)
(Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/how-to-access-JSON-data-in-HTML5-apps. Luettu: 3.5.2016.)

3.4 JQuery-kirjaston Ajax-funktio

JQueryn Ajax-funktio muodostetaan $.ajax(url[, settings]). Kyseinen funktio on lähteiden perusteella eräänlainen ”power-function” monien muiden kirjaston Ajax-funktioiden joukossa, joilla on spesifimmät käyttötarkoituksensa. Ajax-funktion settings-osiossa on mahdollista hienosäätää lukuisa joukko määrityksiä kyselyn toiminnallisuuden ohjaamiseksi, aina kyselyn metodeista tietotyyppeihin, virheenkäsittelijöihin sekä default-arvojen muuttamiseksi sekä huomattava joukko muitakin ominaisuuksia. Dokumentaatio sekä funktion toiminnallisuus on melko laaja kokonaisuus kaikkine poikkeuksineen ja käyttötapauksineen. Toisaalta, funktion settings-osuus on täysin valinnainen eli aivan peruskäytössä funktio toimii myös muodossa $.ajax(url). Tekeillä olevassa mobiiliohjelmassa jQueryn Ajax-funktiossa on käytetty kuitenkin seuraavia asetuksia, joidenka toiminnasta (erityisesti dataType:'json' sekä succes handler) on kerrottu jQueryn Ajax-funktion dokumentaatiossa sekä liitteessä 5.:

type:'GET' (metodi),
url:'http://xxxxxx.com/xxxx.php', (kyselyn osoite)
dataType:'json', (palauttaa JSON-datasta objektit)
success: function (data), (succes handler)

(W3Schools. Table of all Ajax-functions. Luettavissa:
http://www.w3schools.com/jquery/jquery_ref_ajax.asp. Luettu: 3.5.2016.)
(JQuery Ajax-funktion dokumentaatio. Luettavissa: http://api.jquery.com/jquery.ajax/ Luettu: 3.5.2016.) (Liite 5. Ote jQueryn Ajax-funktion dokumentaatiosta, Data Types)

3.5 JQueryn Ajax-funktio ja ”Same-Origin-Policy”

Mobiiliohjelman kannalta, ja käytettäessä jQueryn Ajax-funktiota, on em. funktion dokumentaatiossa mielenkiintoinen kohta, jossa todetaan että mm. JSONP-pyyntö (JSON PADDED) ei olisi tämän ”Same-Origin-Policy”:n rajoitusten piirissä lainkaan.
(JQuery Ajax-funktion dokumentaatio: http://api.jquery.com/jquery.ajax/ Luettu: 3.5.2016.)
(Liite 1. Ote jQueryn dokumentaatiosta, JSONP-pyyntö ja same-origin-policy)

3.6 Intel XDK:n dokumentaatio ja ”Same-Origin-Policy”

Intel XDK -kehitysalustan dokumentaatio tarjoaa ensin ratkaisuksi same-origin-policy:n sivuuttamiseksi kolmea esimerkkikoodia joissa kaikissa on perusfunktiona $.getJSON.
(Liite 2. Ote Intel XDK:n dokumentaatiosta, same-origin-policy ja esimerkkikoodit)

Edellisten lisäksi on tarjolla vielä yksi vaihtoehto, joka jättää kehittäjälle vapaammat kädet päästä käsiksi palvelimen JSON-dataan ja joka soveltuu myös tilanteisiin joissa palvelin ei tue JSONP (JSON PADDED) -formaattia tai ei ole mahdollista (tai ei haluta mm. tietoturvasyistä) käyttää dokumentaatiossa mainittua, headerissa välitettävää Acces-Control-Allow-Origin -tapaa: Intel XDK:ssa same-origin-policy voidaan kiertää liittämällä ohjelman resursseihin (head-tagiin) cordova.js-kirjasto.
(Liite 3. Ote Intel XDK:n dokumentaatiosta: Accessing JSON data in HTML5)
(Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/how-to-access-JSON-data-in-HTML5-apps. Luettu: 3.3.2016.)

Toisin sanoen liittämällä cordova.js -skripti mobiiliohjelman resursseihin, on mahdollista käyttää jQueryn Ajax-funktiota ”Same-Origin-Policy”:n haittaamatta. Ohjelman rakennusvaiheessa on mahdollista määrittää sallitut verkko-osoitteet. Oletusarvo on ”*”, joka sallii kyselyt kaikkiin osoitteisiin. Mainittakoon, että tämä ei tietenkään ratkaise itse palvelinpuolella olevia mahdollisia muita tietoturvaongelmia.
(Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/how-to-access-JSON-data-in-HTML5-apps. Luettu: 3.3.2016.)

4 Tietokantakysely

Ajaxin käyttö on siis täysin riippumaton palvelinpuolella käytettävästä tekniikasta, joten koska vaihtoehtoja on lukuisia, täytyy ottaa kantaa tässä opinnäytetyössä käytettävään tekniikkaan. (Asleson, R., Scutta, N. T. 2006. 19.)

Ulkoisena palvelinresurssina toimii Apache-palvelin, jossa on tuki PHP:lle sekä tietokantapalvelimena MySql-palvelin. Tietokantakysely toteutetaan PHP:llä ja tulos muokataan PHP:n json_encode -funktiolla JSON-muotoon.

4.1 MySql-tietokantaresurssin kuvaus

Palvelimella on kaupallinen lomakeohjelma eli ns. ”back-end”, jolla tietojen tallentaminen tietokantaan toteutetaan. Lisäksi lomake tallentaa myös kuvan kansioon ja samalla kuvan uuden nimen tietokantaan. Lomakeohjelman tietokanta on relaatiotietokanta, mutta se tallentaa kaikki mobiiliohjelmassa tarvittavat tiedot yhteen tauluun, joten myöhemmin tehtävä SQL-kysely kohdistetaan vain siihen. Tämä tietokanta ja sille tarvittava tietojentallennuslomake olisi helppoa koodata itsekin, mutta tässä toteutuksessa erillinen tallennustyökalu on integroitu osaksi kokonaisuutta. Tässä alempana on kuvaus tietokannasta, johon myöhemmin toteutettava tietokantakysely kohdistetaan.

Tämän ohjelman tarvitsemat kentät lomakkeella ovat järjestyksessä:

1. Company
2. Mustateksti
3. Punainenteksti
4. Tekstikenttä tekstille ”Valid”
5. Päivämäärä
6. Web-site/teksti
7. Edellistä varten erillinen HREF-kenttä, jolla voidaan ohjata klikkaus tarkempaan osoitteeseen sekä välittää parametreja
8. Lopuksi on vielä kuvan lataus

Kyseiset lomakkeen tekstinsyöttökentät näkyvät MySql -kannassa seuraavina kenttinä ja muodostavat yhdessä siis tietokantakyselyssä haluamamme tietueen. Tietue palauttaa myös kuvan nimen ja lisäksi tiedämme myös polun, jossa kuva sijaitsee ja tätä tietoa on mahdollista käyttää myöhemmin itse mobiiliohjelmassa.

Tietokannan kentät järjestyksessä:

1. text_1 (Company)
2. text_2_1 (Mustateksti)
3. text_2_2 (Punainenteksti)
4. text_3 (Tekstikenttä tekstille ”Valid”)
5. text_4 (Päivämäärä, muodossa Y-m-d)

6. text_5 (Web-site/teksti)
7. text_6 (kuvan nimi -kenttä)
8. text_7 (HREF-kenttä)

Lomakkeen kentät ovat eri järjestyksessä kuin tietokannan kentät mikä johtuu siitä, että lomakeohjelma lisää tietokantaan kentät suoraan siinä järjestyksessä kuin lomakeohjelmaan on lisätty uusi syöttökenttä tai muu kontrolli.

4.2 Tietokantakysely PHP:llä

Tässä opinnäytetyössä tietokantakysely toteutetaan assosiatiivisena taulukkona käyttämällä while-silmukassa PHP:n mysqli_fetch_assoc-funktiota. Taulukkoa kutsutaan assosiatiiviseksi, jos sen avaimina käytetään merkkijonoja. Assosiatiivinen taulukko määritellään antamalla array()-komennolle arvojen lisäksi avain. Assosiatiivisen taulukon arvoihin viitataan käyttämällä hakasulkuja ja yksittäisen arvon avainta (samoin kuin numeroin indeksoidussa taulukossa, jossa siis viitteenä käytetään numeroita).
(Suomen virtuaaliammattikorkeakoulu. PHP-perusteet, taulukot. Luettavissa: http://www2.amk.fi/mater/tietotekniikka/php-perusteet/taulukot_11362.html Luettu: 9.5.2015)
(PHP:n mysqli_fetch_assoc-funktion dokumentaatio. Luettavissa: http://php.net/manual/en/mysqli-result.fetch-assoc.php Luettu: 9.5.2016.)

PHP:n pitäisi suunnitelman mukaan siis toteuttaa seuraavat vaiheet :

1. Luodaan yhteysobjekti ja avataan tietokantayhteys: new mysqli(host, käyttäjänimi, salasana, tietokannan nimi)
2. Muodostetaan SQL-kysely
3. While-silmukassa käydään kaikki tietueet läpi ja saadaan tuloksena assosiatiivinen taulukko (mysqli_fetch_assoc)
4. Suljetaan tietokantayhteys
5. Muunnetaan tulos json_encode-funktiolla JSON-muotoon (tuloksena objektitaulukko)

(Liite 4. Tietokantakysely PHP:llä ja json_encode-funktio)

4.3 PHP:n json_encode-funktio

Funktio json_encode on ollut tuettuna PHP:n versiosta 5.2.0 lähtien. Kyseinen funktio palauttaa arvon mukaisen JSON-esitysmuodon: ”json_encode — Returns the JSON

representation of a value”. Funktio palauttaa merkkijonon (string) onnistuessaan ja epäonnistuessaan boolean-arvon FALSE. Merkkijonon data pitää olla UTF-8 merkistökoodattu. (PHP:n json_encode-funktion dokumentaatio. Luettavissa: http://fi2.php.net/manual/en/function.json-encode.php Luettu: 9.5.2016.)

Tässä opinnäytetyössä json_encode funktiota käytetään tietokantakyselyn tuottaman taulukkomuotoisen datan muuntamiseen JSON-muotoon ja se tapahtuu yksinkertaisesti merkitsemällä kyselyn for-silmukan ohjelmalohkon ulkopuolelle ”json_encode($json);” . (Liite 4. Tietokantakysely PHP:llä ja json_encode-funktio)

Lopputuloksena on merkkijono joka on objektien muodostama taulukko alla olevan esimerkin mukaisesti (vrt. 4.1. MySql-tietokantaresurssin kuvaus, kentän nimi ja kenttään tallennettu arvo):

[{”id”:”string”,”text_1”:”arvoA”,”text_2_1”:”arvoB”...},
{”id”:”string”,”text_1”:”arvoC”,”text_2_1”:”arvoD”...}...]

Vastauksesta nähdään että taulukossa jokainen tietue muodostaa aaltosulkeiden sisällä objektin eli kyseessä on objektitaulukko. (Liite 6. Haaga-Helia. Orientaatio ohjelmistotuotantoon, objektitaulukko)

5 JSON-formaatti

JSON on lyhenne sanoista JavaScript Object Notation. JSON on tekstiformaatti joka on kieliriippumaton. JSONin virallisilla sivuilla (json.org) on listattuna kymmeniä ohjelmointikieliä joita se tukee. (Asleson, R., Scutta, N. T. 2006, 31, 69)

JSON on rakennettu kahden tietorakenteen varaan, jotka ovat käytännössä kaikkien modernien ohjelmointikielten tukemia. Tietorakenteet ovat:

1. Nimi/arvo-parien joukko (ohjelmointikielissä olio, tallenne tai sanalista)
2. Arvojen järjestetty lista, joka on useimmiten toteutettu taulukkorakenteena

Kyseiset rakenteet löytyvät useimmista ohjelmointikielistä, joten JSON on ideaali vaihtoehto erilaisten järjestelmien välillä tapahtuvaan tiedonsiirtoon. JSON pohjautuu myös JavaScript-standardin osajoukkoon, jolloin sen käsittelyn pitäisi olla yhdenmukaista selaimista riippumatta. Lisäksi (mm. Ajaxia käytettäessä) se on myös vaihtoehto XML-kielelle. (Asleson, R., Scutta, N. T. 2006. 69, 70)
(Introducing JSON. Luettavissa: http://www.json.org/ Luettu: 9.5.2016)

5.1 JSON-formaatin hyödyt tietokantakyselyssä

JSON-formaatti on kevytrakenteinen, tiedonvaihtoon tarkoitettu formaatti ja se on selvästi XML-formaattia tiiviimpi. Pienempi koko voi johtaa suuriin suorituseroihin, kun tietoverkossa siirretään suuria määriä tietoa. Parempi suorituskyky johtaa lyhyempiin vasteaikoihin ja tunnetusti parantaa käyttäjäkokemusta.
(Asleson, R., Scutta, N. T. 2006. 71.)

6 Palvelinpuolen turvallisuudesta

Yleisenä tietona voidaan todeta että palvelinpuolen turvallisuuden varmistaminen on laaja aihe ja jatkuvan kissa-hiiri leikin alaisena. Turvallisuuskysymyksiä ei voi koskaan ohittaa kun rakennetaan sovelluksia, joihin tavalla tai toisella on yleisöllä pääsy. Tämän tästä saa lukea uutisia joissa tunnettujenkin verkkopalvelujen tietovarastoihin on onnistuttu murtautumaan ja käyttäjiä kehoitetaan mm. vaihtamaan salasanansa.
Tietokantapohjaisissa sovelluksissa esiin nousevat tietokantapalvelimen turvallisuuden varmistaminen ja tunnetut tavat yrittää murtautua niihin.

Alla englanninkielisestä lähteestä poimitut suojautumiskeinot eivät kata tietokantapalvelimien turvallisuuskysymyksiä läheskään täydellisesti, mutta hyvänä peruslähtökohtana on aina tarkastaa käyttäjän syöte.

Riippumatta siitä, käytetäänkö tietokantasovelluksessa GET tai POST menetelmää, on käyttäjän syöte aina tarkastettava.
(Nixon R. 2012. Learning PHP, MySQL, JavaScript, and CSS : a step-by-step guide to creating dynamic websites. Sähkökirja. O'Reilly Media. Sebastopol (Calif.), 511)

Käyttäjät voivat yrittää syöttää sovellukseesi mm. haitallista JavaScriptiä tai suoraan MySql-käskyjä murtaakseen tietokantasi turvallisuuden. (Nixon R. 2012. 511)

Tunnettuja tapoja yrittää murtautua tietokantaan ovat esim. Cross-Site-Scripting (XSS) sekä SQL-injektiot. (Nixon R. 2012. 479, 484)

Keinoja suojautua näitä perustapauksia vastaan, on aina tarkastaa käyttäjän palvelimelle tarjoama syöte, riippumatta metodista tai väylästä josta syötettä tarjotaan. PHP tarjoaa useita valmiita funktioita tähän tarkoitukseen, joista voidaan mainita ainakin mysql_real_escape_string, stripslashes, htmlentities ja strip_tags.
(Nixon R. 2012. 481, 485-486, 488, 511-512)

Lisähuomautuksena todettakoon että edellisen kappaleen funktioille on tullut myös uudemman tietokantayhteyskäytännön, kuten mysqlin, mukaiset funktiot, joidenka funktio-ominaisuudet ovat hieman kehittyneet ja poikkeavat siten hieman edellisistä.

7 Ajax-vastauksen käsittely JavaScriptillä

Viimeisenä vaiheena on käsitellä mobiilohjelmassa Ajax-funktion vastaanottama JSON-data JavaScriptillä siten että puhelimen ruudulle ilmestyy esitettävä sisältö. JSON-objektitaulukon objektit, jotka on muodostettu tietokannan tietueista, käydään for-silmukassa läpi. Samalla niihin liitetään esittämisessä tarvittavat HTML-elementit. Koodin toimintaa havainnollistava JavaScript-esimerkkikoodi Haaga-Helian Orientaatio ohjelmistotuotantoon kurssilta on liitteenä. (Liite 7. Haaga-Helia. Orientaatio ohjelmistotuotantoon – JavaScript for-silmukka)

For-silmukka saa JSON-objektitaulukosta, paitsi kaikki mainosten esittämisessä tarvittavat tekstitiedot, niin myös kuvan nimen, minkä lisäksi kuvan nimeen liitetään jokaisella kierroksella sen esittämisessä tarvittava HTML-elementti (<img src=”http://www.esimerkkiosoite.fi/kansio/”>), joka sisältää suoran viittauksen palvelimella sijaitsevaan kuvakansioon. Lopputulos viimeistellään erillisellä CSS:llä. (Liite 8. Ohjelmassa käytetty for-silmukka)

On tärkeää huomata että tässä toteutuksessa tietueiden esittämisjärjestys puhelimen ruudulla määrätään jo palvelimella, jossa PHP:llä muodostetaan JSON-objektitaulukko, liitteen 4 mukaisesti.

8 Alustava suunnitelma ja toimenpiteet

Ideana on siis toteuttaa hybridi mobiiliohjelma, joka hakee dynaamisesti päivittyvän tiedon ulkoisella palvelimella sijaitsevasta relaatiotietokannasta, ja näyttää tuon tiedon halutussa muodossa, kun mobiiliohjelmaa käytetään. Mobiiliohjelmien kehitysalustoja löytyy useita, mutta päädyin valitsemaan Intel XDK:n, koska sen käyttämä tekniikka on itselleni ennestään tuttu. Itselläni on myös kaupallinen palvelimella toimiva lomakeohjelmisto (back-end) jolla päivitettävän tiedon, sekä kuvien lisääminen tietokantaan sujuu kätevästi.

Välttämättöminä esitoimenpiteinä on siis asentaa kehitysalusta, Intel XDK, tutustua sen toimintaan, sekä suunnitella ohjelman tarvitsema tietokanta, sekä tarkistaa lomakeohjelman toteuttama tietokantarakenne eli tietokantakyselyssä halutun tietueen SQL-parametrit.

Mobiiliohjelman ja ulkoisen relaatiotietokannan välisen yhteyden toteuttamiseksi hyödynnetään tekniikkaa, joka on yhdistelmä Ajaxia, PHP:tä, SQL:ää, PHP:n yhtä monista JSON-funktioista sekä edelleen JavaScriptiä, jolla haluttu tieto saatetaan mobiiliohjelmassa esitettävään muotoon. Lopullinen ulkoasu esitettävälle tiedolle viimeistellään lopuksi CSS:llä.

Tutustuessani edellämainittuun tekniikkaan, sain huomata että koodaaminen on erittäin luova laji, jossa tämänkin toteutuksen lopputulokseen pääsemiseksi on tarjolla lukuisia vaihtoehtoja variaatioineen. Esittelen tässä siis yhden tavan, joka noudattelee soveltuvin osin mm. Haaga-Helian kurssimateriaalia ”Orientaatio ohjelmistotuotantoon” -kurssilta sekä nykyaikaista tapaa käyttää JavaScriptin kirjastoja ohjelman toteuttamiseen.

Kirjaston kirjavalikoima tarjosi melko vähän ajankohtaista tietoa aiheesta, joten opintomateriaalina käytin myös mm. jQueryn, Intel XDK:n, PHP:n ja JSONin virallisia www-sivuja sekä dokumentaatioita. Lisäksi kävin suuren määrän koodareille suunnattuja foorumeja läpi kun etsin parhaita käytäntöjä tämän kokonaisuuden toteuttamiseen.

8.1 Toteutus

Kokonaisuus on toteutettu prototyyppivaiheessa seuraavalla tavalla, mutta kehittyneempi versio on jo tekeillä. Intel XDK on mm. ilmoittanut prototyypin valmistumisen jälkeen, että se ei jatkossa tue jQueryn Mobile-kirjastoa (mitä käytettiin prototyyppivaiheessa) ja turvallisuuteen, ohjelman ulkonäköön sekä toimintaan on tullut uusia ideoita.

Nyt Google Playssa julkaistu versio on toteutettu uudestaan käyttämällä Intel XDK:n Ionic-kirjastoa. Lisäksi ohjelman käyttämä tietokanta ja sen tallennusohjelma on asennettu kaupalliseen webhotelliin jossa se toimii täsmälleen samoin kuin kotiserveriltäni.

Intel XDK:n kehitysalusta on saatavilla Windows ja Linux käyttöjärjestelmille ja aloitin asentamalla ne molemmille alustoille nähdäkseni että ovatko ne täysin samanlaiset. Ohjelmat ovat molemmilla alustoilla identtiset. Päädyin valitsemaan Linux-asennuksen koska kotiserverissäni on Linux-perheeseen kuuluva käyttöjärjestelmä ja on helpompi operoida yhdellä koneella, jolla on samalla kertaa pääsy kaikkiin resursseihin. Intel XDK:n käyttäminen edellyttää rekisteröitymistä palvelun käyttäjäksi, joten asennusvaiheessa Intelin palveluun oli luotava tili. Kotiserverilläni on kaupallinen lomakeohjelma, joka tallentaa lomaketiedon MySql-kantaan ja myös kuvan kansioon. Mobiiliohjelmassa näytetään lomakkeelta tallennettu tietuekohtainen data, sekä kyseiseen tietueeseen liittyvä kuva. Lisäksi tarvitaan Android-puhelin johon on ladattu Google Playsta Intelin App Preview-ohjelma ja jonka voi kytkeä usb-johdolla tietokoneeseen ohjelman testausta

varten. Puhelimen asetuksista laite piti asettaa myös ns. developer-modeen, jotta yhteys ohjelman kanssa toimii. Kun ohjelma oli asennettu ja puhelin kytkettynä developer-modessa koneeseen, tein pienen harjoitusprojektin testatakseni että kehitysympäristö toimii.

Seuraavaksi suunnittelin lomakepohjan kaupallisella tietokantapohjaisella lomakeohjelmallani, joka tallensi kantaan kaikki mobiiliohjelman tarvitsemat tiedot sekä tietueeseen liittyvän kuvan kansioon. Tämän jälkeen tutkin phpMyAdminilla tietokannan rakennetta ja minkä nimisiin kenttiin tiedot tallentuvat, sekä mikä oli niiden tietotyyppi. Kaikki tarvittavat tiedot tallentuivat tietokannan yhteen tauluun ja tietotyypeiltään merkkijonoina. Näitä tietoja tarvitsin PHP:llä toteutettuun SQL-tietokantakyselyyn. Vaikka vastaavan tallennustyökalun tekeminen itse olisi ollut helppoa, niin valmis lomakeohjelma sopi tähän tarkoitukseen erittäin hyvin ja päätin integroida sen osaksi kokonaisuutta.

Intel XDK:lla on helppo rakentaa peruskäyttöliittymä ja lisätä ohjelmaan uusia sivuja tarpeen mukaan. Huomasin kuitenkin pian että ohjelman perustyökaluilla ulkoasun muokkaamismahdollisuudet olivat hyvinkin rajalliset, joten tein taustakuvat PhotoShopilla sekä liitin ne itse kirjoittamallani CSS:llä ohjelmaan. Myöskään sopivaa (ja toimivaa) näkymää tietokannasta esitettävälle datalle ei löytynyt valmiina, joten tein sen itse sekä muotoilin näkymän erillisellä CSS:llä. Intel XDK:n käytössä esiintyi lukuisia ongelmia ja jouduin useasti turvautumaan netissä aihetta käsitteleviin sivustoihin. Esimerkiksi tuon itsetekemäni tietokantatiedon esittämiseen tarkoitettu lohko (div) ei vierittynyt automaattisesti ylä- ja alaheaderin alla, mutta lopulta siihen löytyi yksinkertainen ratkaisu lisäämällä CSS-määrittelyyn rivi: overflow: auto. Myöhemmin tosin huomasin että samaan tulokseen päästään, kun ylä- ja alaheadereiden tila määritetään käyttöliittymässä fixed-muotoon.

Totesin useasti, että ensin kannattaa pyrkiä muotoilemaan ohjelman käyttöliittymän generoimin asetuksin ja vasta kun ollaan päästy niillä lähimmäksi, muotoilla omalla CSS:llä. Jouduin itseasiassa useasti rakentamaan projektin alusta puhtaalta pöydältä koska ohjelma lopetti toimintansa, enkä löytänyt mitään selittävää virhettä. Jäin osittain siihen käsitykseen, että liian pitkälle menevät omat muokkailut, vaikka olivatkin ns. oikein tehtyjä, saattoivat aiheuttaa konfliktin ohjelman oman logiikan kanssa. Monesti totesin saman asian peruuttamalla pari vaihetta aiempaan tilanteeseen, jonka jälkeen ohjelma jälleen toimi.

Tietokantakysely palvelinpuolella toteutettiin PHP:llä ja vastaus piti tuloksena saada myös mobiiliohjelman Ajax-funktion ymmärtämään JSON-muotoon. Koska Intel XDK:n kanssa

oli esiintynyt lukuisia ongelmia mm. ohjelman testiajossa, tein serverille ajaxtest.html tiedoston, jonka avulla pystyin varmistumaan koodin toimivuudesta. Välillä sama koodi ei toiminut mobiiliohjelmassa joka johtui ilmeisesti selittämättömistä yhteysvirheistä ja jonkinlaisista tallennusviiveistä.

Tietokantakyselyn kääntäminen juuri oikeaan JSON-muotoon aiheutti aluksi paljon päänvaivaa. JSON-käännöksen lopputuloksena tuli usein virheilmoitus [Object, Object] eli PHP:n json_encode-funktio sai silmukasta tiedon väärin jäsenneltynä. Tässä vaiheessa jouduin palaamaan kirjatiedon ja netin pariin ymmärtääkseni paremmin miten json_encode-funktio toimii ja miten tietokantakyselystä sekä json_encode käännöksestä tulee ymmärrettävää dataa myös Ajaxille. Lopulta, eri lähteistä koottuna, ratkaisu oli kierrättää tietokannan tiedot while-silmukan kautta assosiatiiviseksi taulukoksi, jonka jälkeen json_encode-funktio tuotti tietueista taulukon sisäisiä objekteja, jotka muodostuivat puolestaan tietuekohtaisista nimi-arvoparien joukosta. Kokeilin useita muitakin ratkaisuja mutta tämä oli ainoa joka toimi. Sain ratkaisulle vahvistusta myös aihetta käsittelevältä keskustelufoorumilta, jossa käytettiin samoja ohjelmaresursseja kuin itselläkin. Tämä Ajax-funktion ymmärtämä muoto eli ns. ”Succesful output” on kuvattu kappaleessa 4.3

Jo ennen kuin tuo tietokantakyselyn tulos oli oikeassa muodossaan, sain tietokannasta tietoa näkymään mobiiliohjelmassa käyttäen jQueryn Ajax-funktion sisällä alert(data)-funktiota, jolla pystyin lähinnä toteamaan että tietokantayhteys sekä Ajax-funktio toimii. Tieto piti kuitenkin saada jäsennellyssä muodossa jotta se voitiin mobiiliohjelmassa kierrättää JavaScriptin for-silmukassa ja edelleen saattaa luettavaan muotoon. Funktio-kutsut liitiin mobiiliohjelman ensimmäisen sivun valikkonäkymään ja kutakin nappia painamalla kutsuttiin erillistä Ajax-funktiota, jolle oli annettu seuraavat määreet JQueryn Ajax-funktion dokumentaation mukaisesti:

type:'GET' (metodi),
url:'http://xxxxxx.com/xxxx.php', (kyselyn osoite)
dataType:'json', (palauttaa JSON-datasta objektit)
success: function (data), (succes handler)

JQueryn dokumentaatiossa dataType:'json' ja sitä seuraava rivi (succes handler) on selitetty liitteen mukaisella tavalla. (Liite 5. Ote jQueryn Ajax-funktion dokumentaatiosta, Data Types)

Koska Intel XDK:n dokumentaation mukaan oli mahdollista käyttää jQueryn Ajax-funktiota sivuuttaen em. same-origin-policy liittämällä ohjelman head-tagiin cordova.js-kirjasto niin

valitsin tämän tavan jatkokehittelyä silmällä pitäen. Ohjelman suorituskyvyn kannalta oli myös huolehdittava kuvien keventämisestä, jotta liikenne palvelimen ja mobiiliohjelman välillä tulisi minimoitua. Tietokantatietoa varten JSON oli myös perusteltu valinta, koska lähteiden mukaan se on kevyt formaatti ja soveltuu, sekä on tarkoitettu järjestelmien väliseen tiedonsiirtoon.

Kun tietokannan tiedot oli saatu mobiiliohjelmalle oikeassa JSON-muodossa niin seuraava vaihe oli saattaa ne haluttuun esitysmuotoon. Olin jo selvittänyt netin kautta että JSONin sisältämät tietueet pitää ajaa mobiiliohjelmassa JavaScript-silmukan kautta, aivan kuten palvelinpuolellakin tehtiin PHP-kielellä. Mobiiliohjelmassa ei siis ole käytettävissä PHP-tulkkia joten silmukka toteutettiin JavaScriptillä. Vaikka netti oli jälleen ratkaisuja pullollaan, niin valitsin ratkaisuksi koulumme Orientaatio ohjelmistotuotantoon -kurssin materiaaleista scriptin, joka soveltui pienillä muutoksilla tähän tarkoitukseen. Kysessä on siis for-silmukka, joka käy Ajax-vastauksen objektitaulukon tiedot läpi sekä kasaa ne jäsennellysti teksti-nimiseen muuttujaan. Tuohon teksti-muuttujaan tulivat mukaan myös tietokantatiedon esittämiseen tarvittavat HTML-elementit ja lopuksi ulkoasu viimeisteltiin erillisellä CSS:llä.

9 Mobiiliohjelman julkaiseminen

Valmiin ohjelman julkaiseminen Google Playssa edellyttää rekisteröintymistä Googlen developer-zoneen ja 25 dollarin rekisteröintimaksun maksamista. Rekisteröinti onnistuu helposti olemassa olevalla Google-tilillä. Samalla pitää hyväksyä Google Playn Kehittäjien ohjelmasäännöt sekä Kehittäjien jakelusopimus. Kyseiset säännöt ja sopimus käsittävät pitkät listat mm. moraalisia näkökohtia, tulonjakoa Googlen ja kehittäjän välillä, sisällön rajoituksia, immateriaalioikeuskysymyksiä, harhaanjohtavan toiminnan ja roskasisällön levittämisen aiheuttamia sanktioita, tietosuojaa ja tietoturvaa koskevia määräyksiä, sovelluksen ominaisuuksista annettavien tietojen laatua, asiakastuen antamisvelvollisuutta, ennakkotietoja siitä miten rikkomuksiin reagoidaan sekä erilaisia vaatimuksia itse sovelluksen laadun suhteen.

Ohjelma on mahdollista julkaista myös suoraan Intel XDK:sta käsin mutta, koska ohjelman ja Googlen vaatimien grafiikoiden ominaisuudet eivät täsmänneet niin valitsin julkaisemisen Googlen developer-zonen kautta.

Ennen julkaisua ohjelmasta on talletettava pakollisia tietoja kuten:

-ohjelman nimi (DISCOUNT GRID)
-tiivistelmä ohjelman käyttötarkoituksesta ja ominaisuuksista

-otettava kantaa ohjelman ikärajoitukseen
-valittava maat joissa julkaisu tapahtuu
-em. asiakastuen sähköpostiosoite sekä myös pakollisena seuraavat nimeltä mainitut grafiikat oikeissa muodoissaan:

-small icon 114 x 114px
-large icon 512 x 512px
-screenshot 480 x 800px, min. 3kpl
-small promo 180 x 120px
-large promo 1024 x 500px
-TV-banner 1280 x 720px

Lisäksi voi halutessaan käyttää large promo-kuvan tilalla YouTube-esittelyvideota, josta ilmoitetaan sen osoite ko. palvelussa. Lopuksi ladataan Intel XDK:n tuottama valmis apk-tiedosto palvelimelle ja painetaan julkaise-nappia. Itselläni tuohon käsittelyyn ja julkaisuun meni Googlen osalta n. 4 tuntia, jonka jälkeen sain ilmoituksen sähköpostiini ohjelman julkaisusta.

Ohjelma on ladattavissa Google Playssa nimellä DISCOUNT GRID osoitteessa: https://play.google.com/store/apps/details?id=com.discountgrid.pennitools

Lisäksi valmiin ohjelman kuvat ovat liitteenä. (Liite 9. Kuva 2. Kuvat valmiista DISCOUNT GRID mobiiliohjelmasta)

10 Pohdinta ja tutkimuksen arviointi

Produkti-tyyppisen opinnäytetyöni lopputuotteena syntyi alkuperäistä suunnitelmaani vastaava mobiiliohjelma, jonka ominaisuuksiin ja ulkoasuun olen melko tyytyväinen, ottaen huomioon että kyseessä on ensimmäinen toteutukseni. Koulussa opituista asioista oli paljon hyötyä, mutta itsenäisen tiedonhaun ja soveltamisen merkitys oli tässä produktissa korostunut. Jälkeenpäin, oppimisen kannalta totesin hyödylliseksi sen, että panostin aikaa lähteiden lukemiseen, luin aiheen ulkopuoleltakin eri lähteistä sitä sivuavia artikkeleita ja kokeilin monien esimerkkikoodien toimintaa, jotka eivät suoraan liittyneet itse toteutettavaan mobiiliohjelmaan. Ilman tätä parempaa perehtymistäkin mobiiliohjelma olisi varmasti valmistunut, mutta kokonaisuuden ymmärtäminen sekä oppiminen olisi jäänyt selvästi heikommaksi. Aion luultavasti jatkossakin soveltaa tätä samaa oppimistekniikkaa missä siis ensin perehdytään huolellisesti ja sitten tehdään.

Olen tämän ensimmäisen ohjelman jälkeen tehnyt jo parikin valmista mobiiliohjelmaa ja voi sanoa että innostuin tästä aihealueesta opinnäytetyön kylkiäisenä entistä enemmän. Jatkosuunnitelmissa on hyödyntää eri tavoin tietokantoja mobiiliohjelmakehityksessä sekä mm. avointa dataa, jonka tarjonta lisääntyy ympäri maailmaa jatkuvasti. Lisäksi tarkoituksena olisi perehtyä myös natiivipuolen kehitysalustoihin.

Tämän opinnäytetyöprosessin aikana huomasin useita kehityskohteita, joihin olisi mahdollisuus kiinnittää enemmän huomiota. Mobiiliohjelmien tarjonta on valtavaa, joten ominaisuuksiltaankin kilpailukykyisen ohjelman markkinointiin pitäisi kiinnittää erityistä huomiota. Markkinoinnin osana, ohjelman ulkoasu ja siitä esitettävät grafiikat sekä esittelyvideot, tulisi olla erittäin viimeisteltyjä. Ilman toimivaa markkinointia ja esim. jakamismahdollisuutta sosiaalisessa mediassa (esim. social sharing button), sovellus helposti ”hukkuu” satojen tuhansien ohjelmien joukkoon. Asiakas-palvelin tyyppisissä toteutuksissa palvelinpuolen tietoturvaan kannattaisi myös panostaa, jottei mahdollisesti kovalla vaivalla suosiota saavuttanut ohjelma lopeta yllättäen toimintaansa.

Omassa toteutuksessani ansaintamalli liittyy puhtaasti affiliate-markkinointiin, mutta muitakin tapoja ja niiden yhdistelmiä on tarjolla: ohjelman maksullisuus, ohjelman ominaisuuksiltaan laajemman version maksullisuus, ohjelman käyttöperiodiin tai käyttösisältöön perustuva maksullisuus sekä erilaiset mainostamiseen liittyvät sopimukset. Ansaintamalli voi olla luova yhdistelmä edellisistä tai mobiiliohjelma voi tarjota huomattavaa lisäarvoa jollekin muulle maksulliselle tuotteelle tai palvelulle, esimerkkinä vaikkapa D-Linkin kotivalvontakamera jonka kuvaa voi katsoa puhelimestaan kun lataa siihen ilmaisen, valmistajan tarjoaman Mydlink-ohjelman.

Opinnäytetyöni aihevalinta vaikutti aluksi melko työläältä ja oman osaamiseni rajoja koettelevalta mutta päätin kuitenkin kokeilla vanhaa, hyväksi havaitsemaani tapaa, eli tehdä kattavan, hyvän suunnitelman ja ratkaista eteen tulevat ongelmat vaihe kerrallaan. Opinnäytetyöprosessi oli itselleni ajoittain haastava ja aikataulullisesti se ei olisi ollut kaupallisesti perusteltavissa mutta tietokantapohjaisen, seuraavan ohjelmani toteutukseen menikin enää vain 2 päivää.

Itselleni tämä koko urakka oli hyödyllinen monella tapaa. Arvostan todennäköisesti jatkossa enemmän parempaa etukäteisperehtymistä, sain erittäin hyvän käsityksen mobiiliohjelmiin liittyvästä liiketoiminnasta, eri kehitysalustoista sekä toteutustekniikoista, pystyin hyödyntämään aiempaa osaamistani ja pääsin sisään tämän hetken digitaalisen maailman ehkä yhteen tärkeimmistä kehitysalueista.

Eräs hieno havainto oli, että maailma on täynnä auttamishaluisia koodareita jotka vastaavat esittämääsi avunpyyntöön. Vaihdoin ajatuksia ongelmakohdista mm. erään espanjalaisen sekä parin kiinalaisenkin kanssa. Tietämys on nykyään pitkälti yhteistä ja tulen varmasti vastavuoroisesti jakamaan myös omia kokemuksiani sekä osaamistani kaikille tarvitseville.

Lähteet

Asleson, R., Scutta, N. T. 2006. AJAX – tehokas hallinta. Gummerus. Jyväskylä.

Intel XDK:n lisenssiehdot. Luettavissa: https://software.intel.com/xdk/articles/terms-and-conditions-for-intel-xdk. Luettu: 24.4.2016.

Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/lp-index. Luettu: 25.4.2016.

Intel XDK:n dokumentaatio. Luettavissa: https://software.intel.com/en-us/xdk/docs/how-to-access-JSON-data-in-HTML5-apps. Luettu: 3.3.2016.

Introducing JSON. Luettavissa: http://www.json.org/ Luettu: 9.5.2016

JQuery Ajax-funktion dokumentaatio. Luettavissa: http://api.jquery.com/jquery.ajax/ Luettu: 3.5.2016.

Microsoft. XMLHttpRequest object. Luettavissa: https://msdn.microsoft.com/fi-fi/library/ms535874%28v=vs.85%29.aspx. Luettu: 2.5.2016.

Mozilla Developer Network. Using XMLHttpRequest in IE6. Luettavissa: https://developer.mozilla.org/en-US/docs/Web/API/XMLHttpRequest/Using_XMLHttpRequest_in_IE6. Luettu: 2.10.2016

Nixon R. 2012. Learning PHP, MySQL, JavaScript, and CSS : a step-by-step guide to creating dynamic websites. Sähkökirja. O'Reilly Media. Sebastopol (Calif.).

Suomen virtuaaliammattikorkeakoulu. PHP-perusteet, taulukot. Luettavissa: http://www2.amk.fi/mater/tietotekniikka/php-perusteet/taulukot_11362.html. Luettu: 9.5.2015

PHP:n json_encode-funktion dokumentaatio. Luettavissa: http://fi2.php.net/manual/en/function.json-encode.php Luettu: 9.5.2016.

PHP:n mysqli_fetch_assoc-funktion dokumentaatio. Luettavissa: http://php.net/manual/en/mysqli-result.fetch-assoc.php Luettu: 9.5.2016.

W3Schools.Table of all Ajax-functions. Luettavissa:
http://www.w3schools.com/jquery/jquery_ref_ajax.asp. Luettu: 3.5.2016.

W3Schools. AJAX Inroduction. Luettavissa:
http://www.w3schools.com/ajax/ajax_intro.asp. Luettu: 30.4.2016.

Liitteet

Liite 1. Ote jQueryn dokumentaatiosta, JSONP-pyyntö ja same-origin-policy

Due to browser security restrictions, most "Ajax" requests are subject to the same origin policy; the request can not successfully retrieve data from a different domain, subdomain, port, or protocol. Script and JSONP requests are not subject to the same origin policy restrictions.

Liite 2. Ote Intel XDK:n dokumentaatiosta, same-origin-policy ja $get.JSON sekä esimerkkikoodit

Accessing JSON data from a website is straight forward as long as the data is served from the same domain. However accessing data from a different domain is prohibited by most web browsers because of the Same Origin Policy, which is a security policy that only allows accessing DOM originating from same domain and prevents accessing different domains.

- There are ways you can work around the Same-Origin Policy and access JSON data from different domain from a web application if the server supports one of the following methods:
- JSON response header has Access-Control-Allow-Origin set to allow access to the domain: this is not very common since it poses a security risk. More information here.
- JSON supports padding (JSONP): This is more commonly used by APIs to allow developers to access their data.

JSON data can be accessed from a HTML5 application by making XMLHttpRequest request, or most commonly done using jQuery methods **$.ajax()** or **$.getJSON()**.

1. **function doJSON1**() {

```
$.getJSON('http://time.jsontest.com/?alloworigin=false&callback=?', function (data)
{
    alert(JSON.stringify(data))
});
}
```

2. **function doJSON2**() {

```
$.getJSON('http://time.jsontest.com/?alloworigin=true', function (data) {
    alert(JSON.stringify(data))
});
```

```
}
3. function doJSON3() {
    $.getJSON('http://time.jsontest.com/?alloworigin=false', function (data) {
      alert(JSON.stringify(data))
    });
        }
```

Liite 3. Ote Intel XDK:n dokumentaatiosta: Accessing JSON data in HTML5 hybrid apps using the Intel® XDK

Accessing JSON data in HTML5 hybrid apps using the Intel® XDK

The Intel® XDK is a **HTML5 cross platform app development tool** that lets you write apps using web technologies (HTML5 + CSS + JavaScript*) and build apps that can be installed on Apple iOS*, Android*, Microsoft Windows Phone*, and many other platforms. For an overview of Intel XDK features, open https://xdk.intel.com/.

Intel XDK built Cordova apps provide a way to access JSON data when the JSON does not support padding or if the JSON response header does not have Access-Control-Allow-Origin. Cordova built apps allows you to make ajax calls with HTTP connections using native code which are not subject to cross-origin restrictions. All the app has to do is include cordova.js scripts in the HTML file. You can also restrict the URLs that the app can be allowed to access by providing a white list of URLs in the "Domain list" in the Intel XDK build settings. By default it is set to "*", which allows access to any URL.

cordova.js actual files are not required, it will be included automatically in when a Cordova app is built with Intel XDK, just the script tags should be included.

If you open the sample code ***(with cordova.js script tag included)*** in Intel XDK and test it on emulator or test on real device using Intel App preview app, you will notice that all 3 types of JSON can be accessed. You can also build a hybrid app for iOS, Android, Window Phone and other platform using Intel XDK build service and test the sample app on device.

The Intel XDK Tabs Overview describes the Intel XDK cross-platform development tools you can use to develop apps based on HTML5/JavaScript/CSS code and build a hybrid apps for iOS, Android, Android Crosswalk*, Windows Phone, and other platforms.

Liite 4. Tietokantakysely PHP:llä ja json_encode funktio

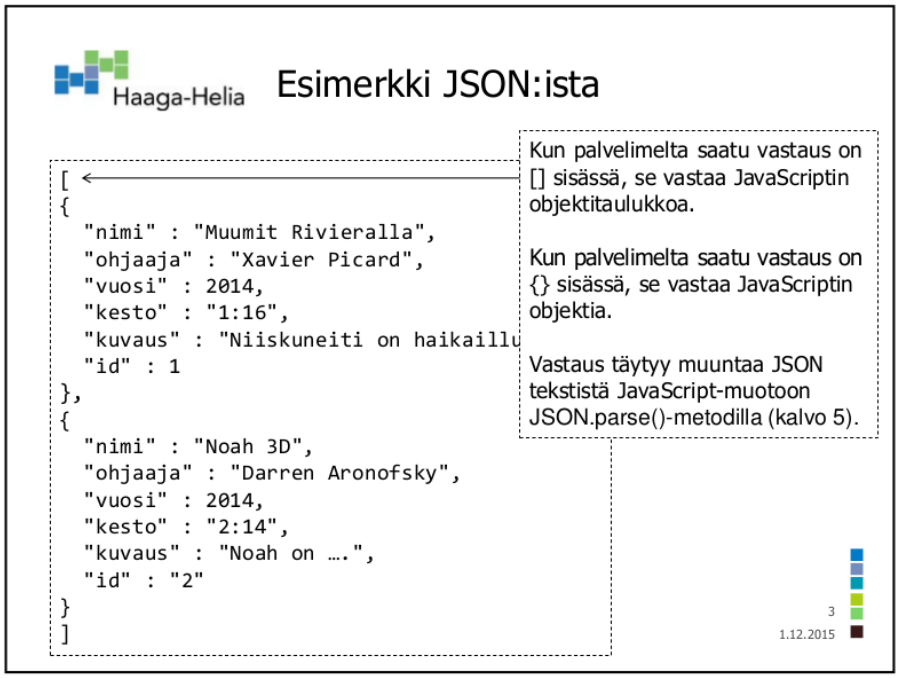
```php
<?php

        $today = date("Y-m-d");

        $mysqli = new mysqli("xxxxxx", "xxxxxxx", "xxxxxx", "xxxxxxxx");

        $query = "SELECT id, xxxxxxx_1, xxxxxx_2_1, xxxxxx_2_2, xxxxxx_3, xxxxxx_4, xxxxxx_5, xxxxxx_6, xxxxxx_7  FROM xxxxxxxx where
        xxxxxx_4 > '$today' ORDER BY id DESC";

        if ($result = $mysqli->query($query)) {
        $json = array();
        /* fetch associative array */
        while ($row = mysqli_fetch_assoc($result)) {

        $json[]=$row;

    }

        /* free result set */
        $result->free();
}
        /* close connection */
        $mysqli->close();
        json_encode($json);
?>
```

Liite 5. Ote jQueryn Ajax-funktion dokumentaatiosta, Data Types

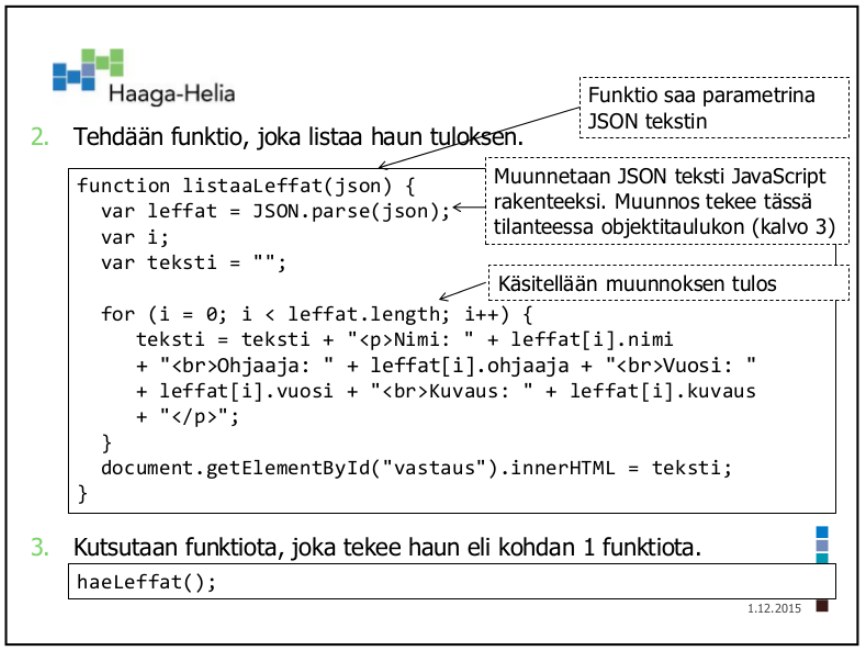
”Data Types

If `json` is specified, the response is parsed using `jQuery.parseJSON` before being passed, as an object, to the success handler. The parsed JSON object is made available through the `responseJSON` property of the `jqXHR` object.”

Liite 6. Haaga-Helia. Orientaatio ohjelmistotuotantoon, objektitaulukko



Liite 7. Haaga-Helia. Orientaatio ohjelmistotuotantoon, JavaScript for-silmukka



Liite 8. Ohjelmassa käytetty for-silmukka

```
var i;
var teksti = "";
var data;

    for (i = 0; i < data.length; i++) {
    teksti = teksti
    +"<div class='outerdiv'>
      <div class='imagediv'>
      <img src='http://xxxxxxx.com/xxx/xxxx/form_xxxxx/files/"
    + data[i].text_6 + "'></div>"
    + "<div class='rightdiv'><p class='companyname'>"
    + data[i].text_1  + "</p>"
    + "<p class='usecode'>" + data[i].text_2_1
    + " " + data[i].text_2_2  + "</p>"
    + "<p class='valid_date'>"
    + data[i].text_3 + "</font>" + "</p>"
    + "<p class='website'>"
    + "<a href='" + data[i].text_5 + "'>"
    + data[i].text_7 + "</a></p></div></div>";

    }
```

Liite 9. Kuva 2. Kuvat valmiista DISCOUNT GRID mobiiliohjelmasta


Intel® XDK Simulator
80%
lenovo
7:22 PM
DISCOUNT GRID
COUPON CODES & GREAT OFFERS
TradingCompass
ALGORITHMIC TRADING SOFTWARE
Free software.
Agorithmic trading system.
Open free account.
CLICK HERE!
PorterFinance
Easy way to trade like a pro
Social trading.
Education center.
Market data.
CLICK HERE!
TRADORAX
FREE WEBINAR
Video library.
Webinars.
Trading Guide.
CLICK HERE!
uBINARY
60 second trading
Trade binary options.
Great training materials.
Open your account now.
CLICK HERE!
OPTION888
Trade binary options
Easy: 1. Choose an asset.
2. Determine the direction.
3. Place your order.
BACK